Vehicle detection and Tracking

Student: Pradyumna Ingle

Q. Explain how (and identify where in your code) you extracted HOG features from the training images. Explain how you settled on your final choice of HOG parameters.

As per [1], Dalal and Triggs received better accuracy on orient 9, Pixels/Cell 8 and Cells/Block 2. I tried this configuration for various channels on all the color channels of YUV, YCrCb, HLS and HSV. Along with that, I calculated time to extract features from training set, train the model, and time taken to evaluate the test set.
I further tried different parameters for Orientation, pixels/cell, cells/block.

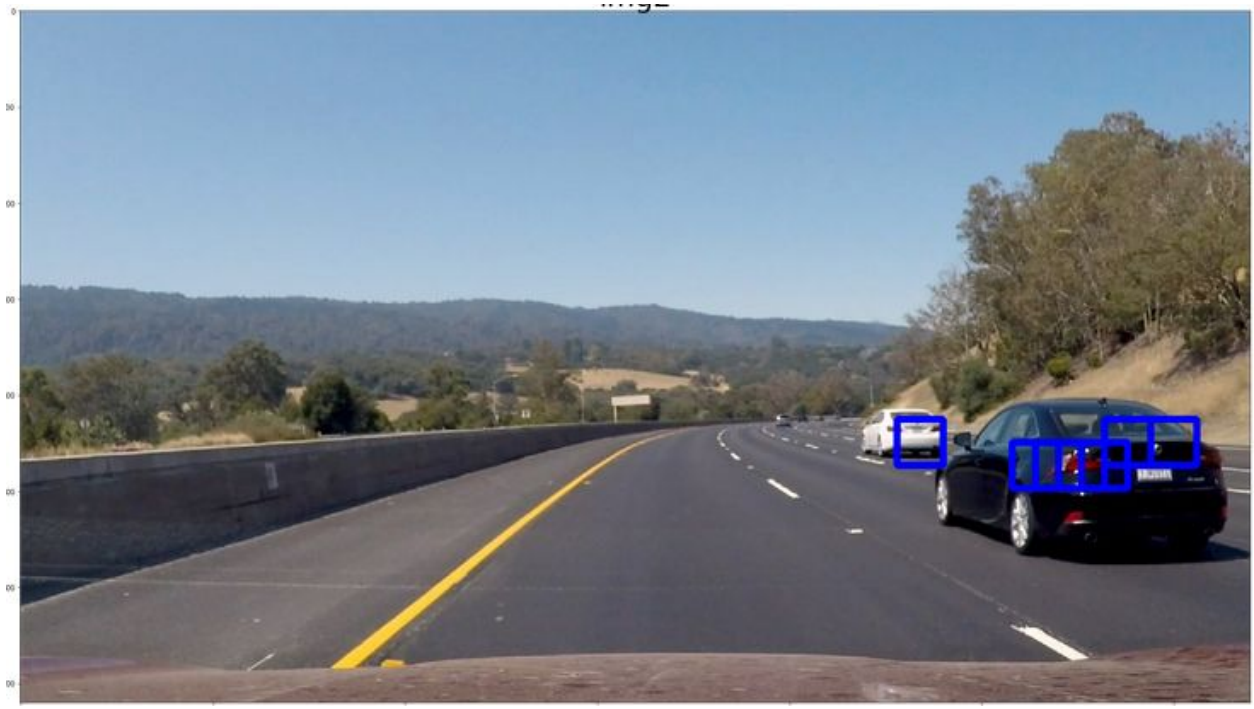
I evaluated result of this extracted data to train Support vector classifier ,Multi-Layer-Perceptrons model and Random Forest model. Not exactly using the Ensemble, I considered output of all three models to give a final decision (Safety is our number one priority (agree that speed was lowered on evaluation)). Hence, the parameters were important to give better results on all three the models. With this consideration as a primary factor, time to evaluate test data, time to extract features, this trial and error attempt was evaluated.

Since Dalal mentions, Gamma correction did not give much correction/improvement, I did not try that.
My final parameters based upon the above criterion are:
Orientations: 16
Pixels/Cell: 16
Cells/Block: 2
Color Channel: YCrCb

Q. Describe how (and identify where in your code) you trained a classifier using your selected HOG features (and color features if you used them).
I used HOG, Histogram and color features to train the model. With that I decided to use Robust Scalar to normalize the data bring all distributions to same scale and overlap.
As mentioned earlier, the Data set was extracted which could give better output on all three models chosen.

Q. Describe how (and identify where in your code) you implemented a sliding window search. How did you decide what scales to search and how much to overlap windows?
Sliding windows were implemented in find_cars() function.
The region considered for finding the cars was the final half i.e. almost from 390 pixels onwards till 650 pixels, given the bonnet/hood of the car in the frame.
Since the longer cars would appear smaller on image, and nearest would appear larger, scale factors used were 0.75, 1, 1.5, 2 and 3. Start and end range vertically is 390 to 670.

Besides for hard mining, I did capture some images of the white car, and road curves to help model train better.



Img: Sliding window for scale 0.75, y_start_end: 425 - 500

Img: Sliding window for scale 1.5, y_start_end: 400 - 650

Img: Sliding window for scale 2, y_start_end: 400 - 650

Img: Sliding window for scale 3, y_start_end: 400 - 650

Q. Show some examples of test images to demonstrate how your pipeline is working. How did you optimize the performance of your classifier?

WIth the sliding window method, each car detection window was given a weightage of 1 on the image with zeroed pixel values (Add_heat()).
The added heat was thresholded to scaled factor of third to the number of windows detecting the cars.
Once that was done, the thresholded heat map was passed through Label(), which binds all the pixels of the same care to same number. That is, all the pixels of the detected car would number same value (label). It returns the list of labels, and the 2d array with labels.

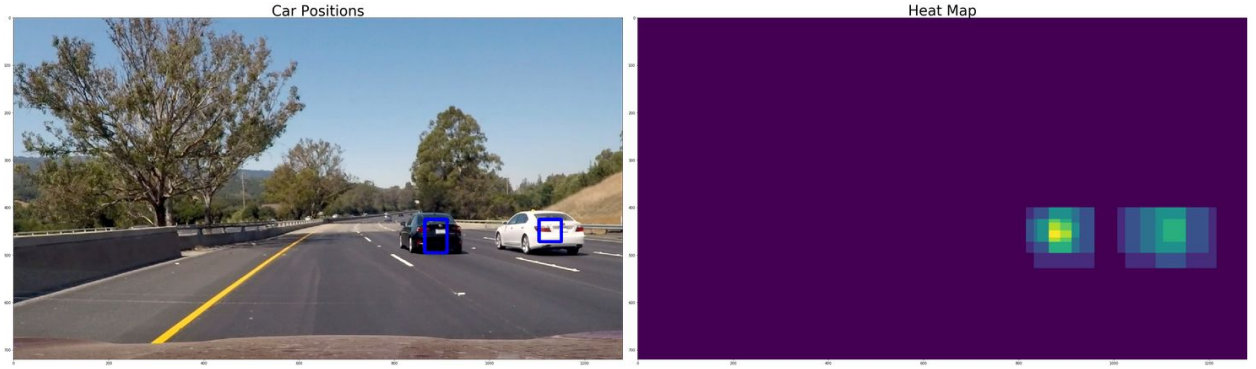
1. I used the labeled 2d array of previous frame and did binary and of the labels with current frame. This helped to mitigate the false positives. Although it is likely loose car detection on first detection frame, since car stays in frames, it starts to fall under detection from very 2nd frame of the car in video.
2. Also, the model does not always detect the cars. In that case, the detection was very infrequent for the same car on the other lanes (white car in video). Just like in the advanced lane detection project, i used data from last 10 frames to keep making the rectangles. Considering avg, video of 30 fps, the car doesn't really change much of it's orientation or direction in ⅓ of second, 10 previous frames should not give much of false negative.
3. Both the approaches to mitigate false positive and false negative are open for debate.
4. Besides above point, for average accuracy of 98% from SVC, we still have ~2% possibilities for error. Considering the safety, I used MLP, SVC and random forest, all three models for predictions. Not sure if this ensembling was a right choice, give it reduced the confidence of the model to the levels of least accurate model, and increased the classification time.

Q. Provide a link to your final video output. Your pipeline should perform reasonably well on the entire project video (somewhat wobbly or unstable bounding boxes are ok as long as you are identifying the vehicles most of the time with minimal false positives.):

1. https://youtu.be/2H2glKzho30 (threshold value is number 1/3rd of total windows finding cars, gamma 1)
2. https://youtu.be/0KjFq3QhWTs (less noisy, by different threshold value, i.e. 3, gamma 0.8)
3. https://youtu.be/kj2nDCHYaD0 (bit noisy, by different threshold value, i.e. 3, no gamma correction)
4. https://youtu.be/xIgxDeZXjMQ (Threshold value, 4, no gamma correction)

Q. Describe how (and identify where in your code) you implemented some kind of filter for false positives and some method for combining overlapping bounding boxes.

As explained above, each rectangle adds weightage of 1 on the pixel value zeroed image. The intensity of the car position increases as multiple overlapping windows detect a car.
now, with appropriately thresholding the image, we can decide for the number of overlapping rectangle boxes to call/consider as a car.
Besides, reducing the gamma of the frame did also help a bit (though gamma adjustments were not done at model training)
The threshold value is very important as it may add false negatives on the perspective.



Img. Heat map and boxes after thresholding (car positions, overall okay threshold)

Img. Heatmap and boxes after thresholding (car positions & noise cancelation)

Img. Heatmap and boxes after thresholding (car positions & False negative for white car)

Img. Heatmap and boxes after thresholding (white car is now bounded after reducing threshold)

Q. Briefly discuss any problems / issues you faced in your implementation of this project. Where will your pipeline likely fail? What could you do to make it more robust?

It actually fails for white car in some frames. I would like to further study the reason. Besides that, how do we trust the classification model that is not 100% accurate on detecting cars/not cars. I felt deciding the threshold value is one of the most difficult part as it affects the false positive and negative.
Using the data from previous frames can/may bring create issues (fraction of seconds delays in taking right decision of breaking.

-- I probably would go for YOLO or Retina Net or faster RCNN to make my decision more robust and gain speed.

Besides, stereo cameras and other sensors (Radar, Lidar) may help better.

References:[1] Histogram of Oriented Gradient
https://lear.inrialpes.fr/people/triggs/pubs/Dalal-cvpr05.pdf